ประพนธ์ อัศวภาณุวัฒน์

SOFT MAIL

Components how to..

เรื่องของการพัฒนาโปรแกรมประเภท Component Objects ซึ่งเป็นสิ่งสำคัญมากสิ่งหนึ่งสำหรับการใช้งานโปรแกรม

สวัสดีครับ ก่อนเข้าเรื่อง ผมมีเรื่องอยากจะบอกกล่าวเกี่ยวกับคำถามยอดนิยมจากผู้อ่าน และผู้ที่จะเริ่มต้นเรียนเขียนโปรแกรมหลายท่าน กับการเลือกภาษาที่จะใช้เขียนโปรแกรม ว่าเลือกใช้ภาษาอะไรดี เป็นคำถามที่ค่อนข้างตอบให้เห็นจริงยากมาก และเป็นคำถามที่อาจก่อให้เกิดการวางมวยกัน เนื่องจากผู้ที่ตอบต่างก็บอกว่าภาษาที่ตนเองใช้เป็นภาษาที่ดีที่สุด (ในโลก) ทำให้ผู้ตั้งคำถามอาจจะไม่ได้คำตอบที่เหมาะสมที่สุด

สำหรับคำถามนี้ผมเองคงตอบให้ไม่ได้แน่ครับ เนื่องจากผมไม่ใช่ผู้เชี่ยวชาญในหลาย ๆ ภาษา แต่มีข้อคิดที่อยากจะฝากไว้ว่า ในปัจจุบันการพัฒนาโปรแกรมแตกต่างจากอดีตอย่างมาก ในอดีตการพัฒนาโปรแกรมจะต้องคำนึงถึงทรัพยากรที่ใช้ในระบบ ทำให้ต้องพยายามเขียนโปรแกรมให้มีขนาดกะทัดรัดมากที่สุด เพื่อที่จะทำงานได้อย่างรวดเร็ว ซึ่งแตกต่างกับปัจจุบัน เรามีเครื่องคอมพิวเตอร์ที่มีคุณภาพสูง มีทรัพยากรในระบบเรียกได้ว่าเหลือเฟือ ทำให้ตัดปัญหาเก่า ๆ ไปได้ จึงไม่จำเป็นที่จะต้องแสวงหาเครื่องมือพัฒนาระบบที่มีความเร็วสูงที่สุด (ที่กล่าวเช่นนี้เนื่องจากความแตกต่างของความเร็วในตัวภาษาคอมพิวเตอร์ต่าง ๆ เราอาจจะไม่สามารถวัดได้ด้วยความรู้สึก กล่าวคือ ต้องใช้โปรแกรมเฉพาะวัดความแตกต่างเท่านั้นถึงจะทราบได้ว่ามีความแตกต่างกัน) และที่สำคัญไม่ว่าภาษาใด ๆ ก็มีข้อดีและจุดด้อยทั้งนั้นครับ แต่สิ่งสำคัญคือ ภาษาใด ๆ ก็ตามเป็นเพียงเครื่องมือในการพัฒนาระบบงานเท่านั้น การที่จะเขียนโปรแกรมออกมาให้ดีพร้อมขึ้นอยู่กับผู้ใช้มากกว่าว่าจะสามารถดึงเอาความสามารถของภาษาออกมาใช้ได้มากเท่าไร ผมขอฝากไว้สำหรับผู้ที่มีคำถามนี้อยู่ในใจครับ

Delphi Components

ฉบับนี้เราจะคุยกันเรื่องการพัฒนาโปรแกรมประเภท Component Objects ซึ่งเป็นสิ่งสำคัญมากสิ่งหนึ่งสำหรับการใช้งานโปรแกรมประเภท Visual-Development Environments การเขียนโปรแกรมประเภท Component Objects มีลักษณะเช่นเดียวกับการเขียนโปรแกรมที่เป็น Class หรือ Objects ทั่ว ๆ ไป แต่สิ่งที่แตกต่างคือ Component Objects จะมีส่วนสนับสนุนสำหรับระบบ User Interface ได้ดีกว่า Objects ทั่ว ๆ ไป ดังที่พบในโปรแกรม Delphi เอง เพื่อไม่ให้เสียเวลาผมขอกล่าวถึงขั้นตอนการสร้าง Component Objects มีดังต่อไปนี้ครับ

เลือก Ancestor Class

การสร้าง Components ส่วนที่สำคัญที่สุดคือการเลือกการสืบสกุลจากคลาสบรรพบุรุษให้เหมาะสม สำหรับ Base Class ใน Delphi ประกอบไปด้วย 5 Class คือ

Tobject เป็น Base Class ของทุก ๆ Class ที่มีใน Delphi Components เราสามารถสืบสกุลจาก Class เพื่อสร้าง Components ที่ไม่ต้องการแก้ไของค์ประกอบใน Object Properties สำหรับ Components ที่สืบสกุลจาก Tobject จะไม่ปรากฏอยู่ใน IDE Pallete

Tcomponent Base Class ชนิด None-visual components กล่าวคือ จะไม่แสดงตัวบนฟอร์มในขณะ Run-time คอมโปเนนต์ที่สืบสกุลจาก class นี้ได้แก่ Ttimer, Tdatasource และ Ttable เป็นต้น

TgraphicControl Base Class ที่เพิ่มเติมส่วนของกิจกรรม Canvas และกิจกรรม Paint ทำให้สามารถแสดงผลใน Objects ที่เรียกใช้ได้ TgraphicControl จะไม่มี Windows Handle จึงไม่สามารถ SetFocus ได้

TwinControl ใช้เป็น Base Class สำหรับเกือบทุก Windows Components มี Windows Controls (Handle) และสามารถ SetFocus ได้

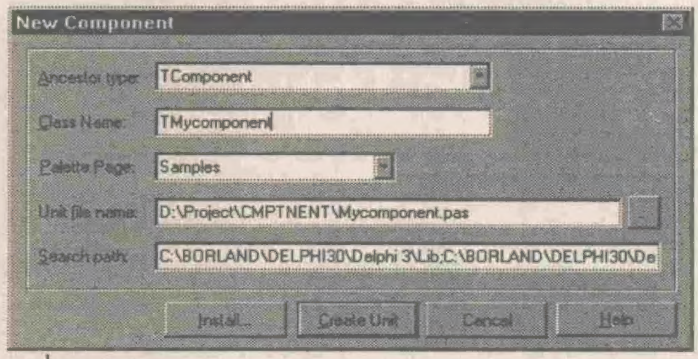
TcustomControl Base Class ที่รวมเอาความสามารถของ TgraphicControl และ TcustomControl เข้าด้วยกัน

นอกจาก Base Class ทั้ง 5 เราสามารถสร้าง Components ให้สืบสกุลจาก Component Objects ตัวอื่น ๆ ได้โดยตรงเช่นกัน

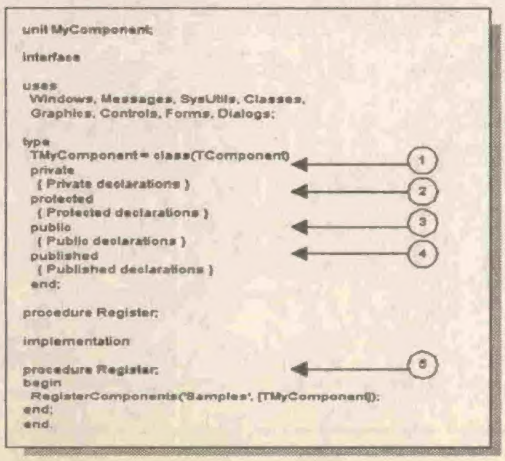
กำหนด Component Framework

โดยปกติเมื่อต้องการสร้าง Components จะสามารถใช้ Delphi Project Wizard โดยเข้าสู่เมนู File —> New และเลือก Components โปรแกรม Delphi จะแสดงหน้าต่าง New Component ดังรูปที่ 1

- ให้เลือก Base Class ที่ต้องการสืบสกุลในช่อง Ancestor Type ตามต้องการ
- ระบุชื่อ Class Name (ขอให้เริ่มต้นชื่อ Class ด้วย "T" เพื่อเป็นมาตรฐานการกำหนดเป็นแบบข้อมูล)
- เลือกหน้า Palette ที่ต้องการบรรจุ Component ใหม่ที่สร้างขึ้น



รูปที่ 1 หน้าต่าง New Component (Component Framework)



รูปที่ 2 Basic Component Source Code

- ระบุชื่อยูนิตในช่อง Unit file Name
- และระบุ Search Path โดยการเพิ่ม Path ที่บันทึกไฟล์ยูนิตไว้
- กดปุ่ม [Create Unit] เพื่อทำการเริ่มสร้างโครงยูนิตดังรูปที่ 2

โครงโปรแกรมของยูนิตที่ใช้สร้างเป็น Component Objects จะเหมือนกับยูนิตที่ใช้สร้าง Object Class ทั่วไป มีส่วนที่สำคัญดังต่อไปนี้

1. Private declarations เป็นส่วนที่ใช้ระบุตัวแปรประเภท Internal Data Field เพื่อใช้บันทึกข้อมูล (Interface Field) และ Internal Data Event เพื่อใช้กำหนด Events ต่าง ๆ (Interface Events) สำหรับ Property และ Events ทุกตัวที่สร้างขึ้น รวมทั้งระบุกิจกรรมและตัวแปรต่าง ๆ ที่ต้องการใช้เฉพาะใน Object นี้

2. Protected declarations เป็นส่วนที่ใช้ระบุกิจกรรมที่ต้องปกป้อง หรือต้องการให้ผู้ใช้คนต่อไปทำการ Override ก่อนนำไปใช้งาน

3. Public declarations ส่วนที่ใช้ระบุกิจกรรม (Method) หรือฟังก์ชัน (Function) ที่ผู้ใช้สามารถนำไปเรียกใช้งานได้

4. Published declarations ส่วนที่ใช้ระบุ Property ชนิด Interface Field และชนิด Interface Events

5. Component Register Part ส่วนการลงทะเบียน Component เข้ากับ Delphi IDE

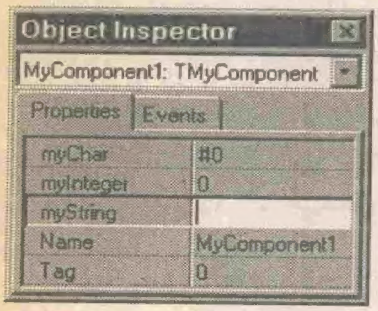
Properties

Properties เป็น Interface field ที่สร้างขึ้นเพื่อให้ผู้ใช้โปรแกรมสามารถติดต่อกับ Components Objects ได้โดยไม่ต้องเข้าถึงชุดข้อมูลภายใน (Internal Data field) โดยตรง ชนิดของ properties จะประกอบไปด้วย แบบข้อมูลทั่วไป แบบข้อมูลชนิด Set ชนิด Object และชนิดอะเรย์ การกำหนดชื่อของ properties ส่วนใหญ่นิยมใช้หลักการตั้งชื่อโดยกำหนดให้ชื่อตัวแรกของตัวแปรที่ตอบสนอง Properties ขึ้นต้นด้วย "F" เช่น Fcolor, Fcaption เป็นต้น

Single Data Field Property

การสร้าง Property ชนิด Single Data Field สามารถกำหนดแบบข้อมูลเป็นตัวเลข ตัวอักษร หรือชุดตัวอักษร (String) Property ชนิดนี้สามารถเปลี่ยนค่าใน Object Inspector ได้โดยตรง ดังรูปที่ 3

```
TMyComponent = class(TComponent)
  private
    { Private declarations }
    FmyInteger    : integer;
    FmyChar       : char;
    FmyString     : string;
  published
    { Published declarations }
    property myInteger : integer read FmyInteger write FmyInteger;
    property myChar    : char read FmyChar write FmyChar;
    property myString  : string read FmyString write FmyString;
end;
```



รูปที่ 3
Single Data Field Property

myInteger เป็น property ที่มีแบบข้อมูลเป็น Integer และมี Read/Write Interface ไปที่ FmyInteger ทำให้ myInteger สามารถอ่านและบันทึกค่าลง FmyInteger ได้โดยตรง เราสามารถบังคับให้ property ใด ๆ มีกิจกรรมการ Read หรือ Write ได้เพียงอย่างเดียว ด้วยการระบุเพียง Read หรือ Write เท่านั้น

Enumerated Data Field Property

Enumerated Property เป็นรายการชุดข้อมูลแบบกำหนดเองสามารถอ่านและเปลี่ยนแปลงค่าโดยตรงได้ใน Object Inspector แสดงผลในรูปแบบ Drop-down listbox ดังรูปที่ 4

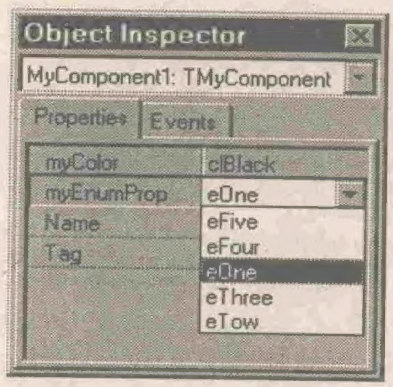
```
type
 TmyEnumProp = ( eOne, eTow, eThree, eFour, eFive );
 TMyComponent = class(TComponent)
 private
   { Private declarations }
   FmyEnumProp   : TmyEnumProp;
   FmyColor      : Tcolor;
 published
   { Published declarations }
   property myEnumProp : TmyEnumProp read FmyEnumProp write FmyEnumProp;
   property myColor : Tcolor read FmyColor write FmyColor;
 end;
```

Set Data Field Properties

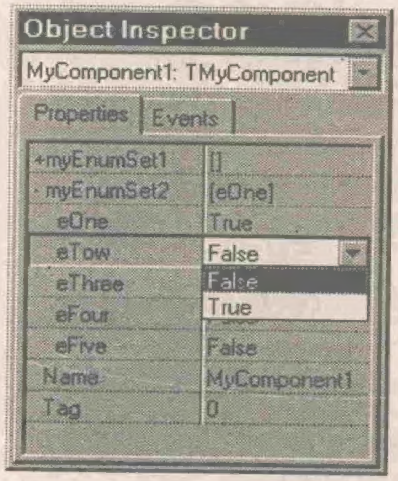
Set Properties เป็นแบบข้อมูลตามแบบ Pascal แสดงอยู่ใน Object Inspector ในรูปแบบเครื่องหมาย + อยู่หน้า Property และมีเครื่องหมายก้ามปู "[ ]"อยู่ในช่อง Property Value โดยมี elements ได้สูงสุด 32 ตัว และแต่ละตัวแทนด้วย True หรือ False เพื่อระบุว่าเป็นสมาชิกของ Set หรือไม่ เราสามารถอ่านค่าและเปลี่ยนแปลงค่าโดยตรงใน Object Inspector ดังรูปที่ 5

```
type
 TmyEnumProp = ( eOne, eTow, eThree, eFour, eFive );
 TmyEnumSet  = Set of TmyEnumProp;

 TMyComponent = class(TComponent)
 private
   { Private declarations }
   FmyEnumSet1 : TmyEnumSet;
   FmyEnumSet2 : TmyEnumSet;
 published
   { Published declarations }
   property myEnumSet1 : TmyEnumSet read FmyEnumSet1 write FmyEnumSet1;
   property myEnumSet2 : TmyEnumSet read FmyEnumSet2 write FmyEnumSet2;
 end;
```



รูปที่ 4
Enumerated Property



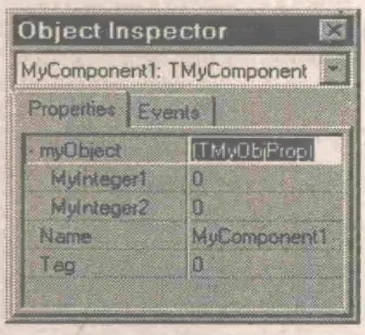
รูปที่ 5
Set Data Field Property

Object Properties

Objects Properties คือ Property ชนิด Class ที่เชื่อมโยงไปยัง Objects หรือ Component อื่น เข้าเป็นสมาชิก แสดงอยู่ Object Inspector ในรูปแบบเครื่องหมาย "+" ข้างหน้า และมีชื่อ Component Objects หรือมีปุ่ม [...] ในช่อง Property Value การแก้ไขเปลี่ยนแปลง Property Value สามารถทำได้โดยตรงใน Object Inspector หรือใช้ Custom Property Editor ขึ้นอยู่กับผู้ออกแบบ Objects เอง สำหรับ Component ที่จะนำมาใช้ต้องสืบสกุลจาก Tpersistent เท่านั้น ดังรูปที่ 6

```
 TMyComponent = class(TComponent)
 Private
   { Private declarations }
   FmyObject : TmyObjProp;
 Public
   { Public declarations }
   constructor Create( AOwner : TComponent ); override;
   destructor Destroy; override;
 published
   { Published declarations }
   property myObject : TmyObjProp read FmyObject write FmyObject;
 end;
```

```
TMyObjProp  = class(TPersistent)
 private
  { Private declarations }
  FmyInteger1 : integer;
  FmyInteger2 : integer;
 published
  { Published declarations }
  property MyInteger1 : integer read FmyInteger1 write FmyInteger1;
  property MyInteger2 : integer read FmyInteger2 write FmyInteger2;
end;
```



รูปที่ 6 Object Property

Array Property

Array Property คือ property ที่สามารถเข้าถึงข้อมูลในรูปแบบ Array แบบภาษา Object Pascal การกำหนดแบบข้อมูลให้กับ Array Property สามารถกำหนดให้อยู่ในรูปแบบข้อมูลมาตรฐานเช่นเดียวกับ Object Pascal Array Property มีสิ่งที่แตกต่างจาก Array ของ Object Pascal คือ

- Array Property สามารถใช้ index เป็นข้อมูลชนิด String แทนข้อมูลชนิด Integer ได้

- Array Property สามารถเข้าถึง (access) ได้ครั้งละ 1 elements เท่านั้น ไม่สามารถเข้าถึงแบบ Object Pascal โดยการเลื่อน Index Pointer ไปชี้ที่ข้อมูลชุดอื่นได้

Array Property ไม่สามารถแก้ไขโดยตรงด้วย Object Inspector เหมือนกับ Property ชนิดอื่น ๆ ได้ การเปลี่ยนแปลงข้อมูลในช่วงเวลา Designed-time จำเป็นต้องสร้าง Customer Property Editor เพื่อใช้แก้ไขข้อมูลในส่วนนี้ ดังรูปที่ 7

```
type
TmyArrayProp = array[ 1..5 ] of char;
TMyComponent = class(TComponent)
 private
  { Private declarations }
  FmyArrayProp : TmyArrayProp;
  function GetDataByInt( index : integer ) : string;
  function GetDataByStr( index : string ) : integer;
 protected
  { Protected declarations }
 public
  { Public declarations }
  constructor Create( AOwner : TComponent ); override;
 published
  { Published declarations }
  property myArrayInt[ index : integer ] : string read GetDataByInt;
  property myArrayStr[ index : string ] : integer read GetDataByStr;
end;
```

รูปที่ 7 Array Property Declaration

Property Directives

Property directives เป็น additional command ที่ใส่เพิ่มในส่วนของการกำหนด Property เพื่อให้มีผลบังคับใช้กับ Property ตามชนิดของ Directives ต่าง ๆ Directive ที่สำคัญมีดังต่อไปนี้

Default Directive โดยปกติ Delphi จะบันทึก Property Value ลงใน Form (*.DFM) ที่เรียกใช้ Property นั้น ๆ การใช้ Default Directive จะเป็นการกำหนดค่า Default Value ที่ไม่ต้องการให้บันทึกลง Form เช่น

```
Property MyInteger : integer read FmyInteger write FmyInteger default 10;
```

จะเป็นการบังคับให้ MyInteger Property ไม่บันทึกค่า 10 ลงไปใน Form ถ้า Property Value มีค่าเท่ากับ 10

NoDefault Directive ใช้ยกเลิกการกำหนด DefaultValue ด้วย Default Directive ส่วนใหญ่จะใช้ยกเลิก เมื่อทำการสืบสกุลจาก Object บรรพบุรุษที่มีการระบุค่า Default Value ไว้ เช่น

```
Property MyInteger Nodefault;
```

Stored Directive เป็น Directive ชนิด boolean ใช้กำหนดการบันทึก Property Value ลงฟอร์ม ถ้ากำหนดให้เป็น True จะเป็นการบันทึก Property Value ลงฟอร์ม และถ้ากำหนดให้เป็น False จะไม่บันทึกค่าใด ๆ เช่น

```
Property MyInteger : Integer read FmyInteger write FmyInteger stored true;
Property MyString : string read FmyString write FmyString stored false;
```

Reading & Writing เป็น Directive ที่ใช้ระบุความสามารถในการเข้าถึงข้อมูล และบันทึกข้อมูลใน Property เช่น ถ้าระบุ Read อย่างเดียว จะทำให้ไม่สามารถบันทึกค่าลง Property นั้น ๆ ได้ เช่น

```
Property MyInteger : Integer read FmyInteger;
Property MyString : string read FmyString write FmyString;
```

Initializing Values

การกำหนดค่าเบื้องต้น (Initializing) ให้กับ property หรือตัวแปรในระบบ สามารถทำได้สองตำแหน่งคือใน Create Constructor และ Loaded Method โดยทั้งสองกิจกรรมต้องทำการ Inherited กิจกรรมต้นสกุลด้วยทุกครั้ง

Methods

Methods คือกิจกรรมชนิด Procedure หรือฟังก์ชันที่สร้างขึ้นเพื่อให้ User สามารถเรียกดำเนินการกับ Components ได้ เราสามารถสร้างกิจกรรมชนิดต่าง ๆ เช่น Standard, Virtual, Dynamic, Message และ Abstract ใช้ภายใน Components ได้ทั้งหมด

Events

Events เป็น Property ชนิดหนึ่งที่มีแบบข้อมูลเป็น Method Pointer เกิดขึ้นจากผลลัพธ์ของการประมวลผลของระบบปฏิบัติการตอบสนองกับ Components ที่ใช้ในโปรแกรม ซึ่ง User สามารถเขียนโปรแกรมตอบสนองเหตุการณ์ต่าง ๆ ที่เกิดขึ้นได้โดยการสร้าง Events Method ขึ้นรองรับเหตุการณ์ชนิดต่าง ๆ ตามแต่ที่ Components เอื้ออำนวยให้ เช่น MouseClick, MouseMove เป็นต้น

การสร้าง Events ให้กับ Components สามารถทำได้ 2 แบบคือ สร้างโดย Overriding Standard Events และสร้างขึ้นใหม่เป็น Custom Events ทำได้ดังต่อไปนี้

Overriding Standard Events

เป็นการสร้าง Events ใหม่ขึ้นภายใต้กิจกรรม Events เดิมของบรรพบุรุษ โดยการยืมความสามารถของ Events ตัวเดิม และเพิ่มความสามารถใหม่เข้าไปในระบบ เช่น

```
TsoundButton = class( Tbutton )
Protected
  Procedure Click; override;
End;
Procedure TsoundButton.Click;
Begin
  Inherited Click;
  MessageBeep( 1 );
End;
```

Create Custom Events

เป็นการสร้าง Events ขึ้นใหม่โดยไม่ได้สืบสกุลจาก Events เดิมของบรรพบุรุษ การสร้าง Events ชนิดใหม่จะต้องทำการสร้างแบบข้อมูลชนิด Method Pointer เพื่อเป็นแบบข้อมูลที่ใช้สื่อสารระหว่าง Components กับ User และกำหนดตัวแปร ให้มีชนิดเป็นแบบข้อมูล Method Pointer รวมทั้งกำหนดให้มี property ขึ้นรองรับตัวแปรที่สร้างขึ้น เช่น

```
TnotifyMessageEvent  = procedure( sender : Tobject; mesgText :
string; mesgCode : integer ) of Object;
TmyControl = classs( Tcomponent )
Private
  FonNotifyMessage : TnotifyMessageEvent;
Publish
  Property OnNotifyMessage : TnotifyMessageEvent read
FonNotifyMessage write FonNotifyMessage;
End;
```

การตั้งชื่อ Event Property นิยมใช้คำว่า "On" นำหน้าเพื่อเป็นการจำแนก Events Property ออกจาก Property ทั่วไปให้ชัดเจน (เป็นประโยชน์ต่อผู้พัฒนา Components เอง)

Component Icon

เราสามารถสร้างรูป Component Icon โดยบันทึกไว้ใน Delphi Component Resource File (.DCR) ในรูปแบบภาพ Bitmap ขนาด 24x24 จุดที่ 16 สี (หรือ 256 สี) และเปลี่ยนชื่อให้ตรงกับ Component Class รวมทั้งตั้งชื่อไฟล์ .DCR ให้ตรงกับไฟล์ที่บันทึก Components ตัวใหม่ มีลำดับการสร้างดังนี้

1) ใช้โปรแกรม Image Editor ของ Delphi ในการสร้าง Delphi Component Resource File เราสามารถรันโปรแกรม Image Editor โดยการคลิกที่เมนู Tools —> เลือก Image Editor จาก Delphi ToolBar

2) ในโปรแกรม Image Editor เลือกเมนู File —> New —> Component Resource File

3) เลือกเมนู Resource —> New —> Bitmap จะเกิดหน้าต่าง Bitmap Properties

4) ให้กำหนดขนาดภาพ Bitmap เป็น 24x24 ที่ 16 สี

5) สร้างภาพตามต้องการโดยดับเบิลคลิกที่ชื่อ Bitmap1

6) เปลี่ยนชื่อ Bitmap1 เป็นชื่อเดียวกับ Component Class

7) บันทึกชื่อไฟล์เป็นชื่อเดียวกับไฟล์ยูนิต Component

Component PhotoType

ผมสร้าง Component เป็น Phototype ที่รวบรวมเอาชนิดของ Property ที่สามารถสร้างได้ทั้งหมดไว้ใน Components Object เดียว รวมทั้งส่วนการ Initialize ตัวแปรและ Properties แสดงอยู่ในรูปที่ 8 และโปรแกรมแสดงอยู่ในลิสติงที่ 1 มีส่วนของโปรแกรมที่สำคัญดังต่อไปนี้ครับ

Constructor Create(Aowner : Tcomponent); override; แสดงการ Overriding Create Constructor เพื่อเพิ่มส่วน Intializing ให้กับ Component

property myInteger : integer read FmyInteger write FmyInteger; เป็น Property ใช้ทดสอบการอ่านและเขียนข้อมูลชนิด Integer แบบ Single data field property

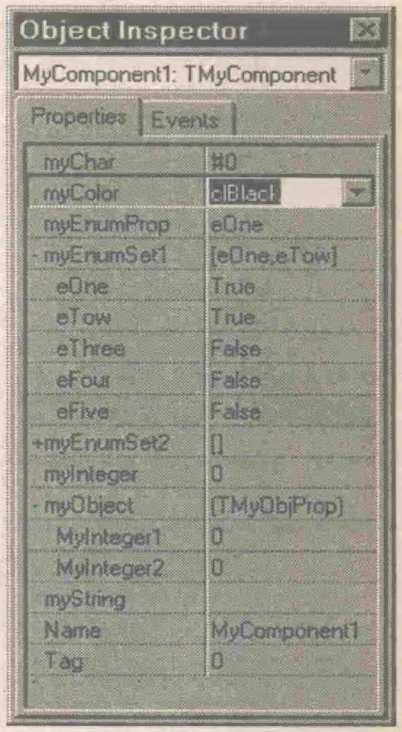
property myChar : char read FmyChar write FmyChar; เป็น Property ใช้ทดสอบการอ่านและเขียนข้อมูลชนิด Char

property myString : string read FmyString write FmyString; เป็น Property ใช้ทดสอบการอ่านและเขียนข้อมูลชนิด String

property myEnumProp : TmyEnumProp read FmyEnumProp write FmyEnumProp; Property แสดงการอ่านและเขียนข้อมูลชนิด Enumerated data field โดยสร้างแบบข้อมูลเอง

ลิสติงที่ 1 Component Phototype

```
00001:  unit MyComponent;
00002:
00003: interface
00004:
00005: uses
00006:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
00007:
00008: type
00009:   TmyArrayProp = array[ 1..5 ] of char;
00010:
00011:  const
00012:  myArrayPropK : TmyArrayProp = ( 'A','B','C','D','E' );
00013:
00014:  type
00015:  TmyEnumProp = ( eOne, eTow, eThree, eFour, eFive );
00016:  TmyEnumSet  = Set of TmyEnumProp;
00017:
00018:  TMyObjProp   = class(TPersistent)
00019:  private
00020:  { Private declarations }
00021:  FmyInteger1 : integer;
00022:  FmyInteger2 : integer;
00023:  published
00024:  { Published declarations }
00025:  property MyInteger1 : integer read FmyInteger1 write FmyInteger1;
00026:  property MyInteger2 : integer read FmyInteger2 write FmyInteger2;
00027:  end;
00028:
00029:  TMyComponent = class(TComponent)
00030:  private
00031:  { Private declarations }
00032:  FmyInteger    : integer;
00033:  FmyChar       : char;
00034:  FmyString     : string;
00035:  FmyEnumProp   : TmyEnumProp;
00036:  FmyColor      : Tcolor;
00037:  FmyEnumSet1 : TmyEnumSet;
00038:  FmyEnumSet2 : TmyEnumSet;
00039:  FmyObject : TmyObjProp;
00040:  FmyArrayProp : TmyArrayProp;
00041:  function GetDataByInt( index : integer ) : string;
00042:  function GetDataByStr( index : string ) : integer;
00043:  protected
00044:  { Protected declarations }
00045:  public
00046:  { Public declarations }
00047:  constructor Create( AOwner : TComponent ); override;
00048:  destructor Destroy; override;
00049:  published
00050:  { Published declarations }
00051:  property myInteger : integer read FmyInteger write FmyInteger;
00052:  property myChar    : char read FmyChar write FmyChar;
00053:  property myString  : string read FmyString write FmyString;
00054:  property myEnumProp : TmyEnumProp read FmyEnumProp write FmyEnumProp;
00055:  property myColor : Tcolor read FmyColor write FmyColor;
00056:  property myEnumSet1 : TmyEnumSet read FmyEnumSet1 write FmyEnumSet1;
00057:  property myEnumSet2 : TmyEnumSet read FmyEnumSet2 write FmyEnumSet2;
00058:  property myObject : TmyObjProp read FmyObject write FmyObject;
00059:  property myArrayInt[ index : integer ] : string read GetDataByInt;
00060:  property myArrayStr[ index : string ] : integer read GetDataByStr;
00061:  end;
00062:
00063:  procedure Register;
00064:
00065:  implementation
00066:
00067:  constructor TmyComponent.Create( AOwner : TComponent );
00068:  begin
00069:  inherited Create( AOwner );
00070:  FmyObject:=TmyObjProp.Create;
00071:  FmyArrayProp:=myArrayPropK;
00072:  end;
00073:
```



รูปที่ 8 Component Property Type

ลิสติงที่ 1 (ต่อ)

```
00074:   destructor TmyComponent.Destroy;
00075:   begin
00076:   FmyObject.Free;
00077:    inherited destroy;
00078:  end;
00079:
00080:  function TmyComponent.GetDataByInt( index : integer ) : string;
00081:  begin
00082:   if ( index in [ 1..5 ] ) then result:=FmyArrayProp[index];
00083:  end;
00084:
00085:  function TmyComponent.GetDataByStr( index : string ) : integer;
00086:  var c : integer;
00087:  begin
00088:   for c:= 1 to 5 do
00089:   begin
00090:   if ( Uppercase(FmyArrayProp[c])=Uppercase(index) ) then
00091:    begin result:=c; exit; end;
00092:   end;
00093:   end;
00094:
00095:   procedure Register;
00096:   begin
00097:   RegisterComponents('Samples', [TMyComponent]);
00098:   end;
00099:   end.
```

property myColor : Tcolor read FmyColor write FmyColor; Property แสดงการอ่านและเขียนข้อมูลชนิด Enumerated เช่นเดียวกัน ใช้แบบข้อมูล Tcolor ของระบบ

property myEnumSet1 : TmyEnumSet read FmyEnumSet1 write FmyEnumSet1; Property แสดงการอ่านและเขียนข้อมูลชนิด Set

property myObject : TmyObjProp read FmyObject write FmyObject; Property แสดงการใช้ Component Objects อื่นเป็น elements

property myArrayInt[index : integer] : string read GetDataByInt; แสดงการใช้งาน array data field โดยใช้ index แบบ Integer

property myArrayStr[index : string] : integer read GetDataByStr; แสดงการใช้งาน array data field โดยใช้ index แบบ string

Share Memory Component

ผมสร้าง Component ขึ้นใหม่เพื่อใช้ทดสอบการเขียนโปรแกรมตามบทความชุดนี้ ชื่อ ShareMem Component จุดประสงค์ของ Component ชุดนี้คือการ allocate หน่วยความจำชนิด Virtual Memory เพื่อใช้บันทึกข้อมูล ข้อดีของหน่วยความจำชนิดนี้คือไม่จำกัดขนาด (ขนาดของหน่วยความจำจะเท่ากับขนาดของพื้นที่ฮาร์ดดิสก์ที่คุณใช้เป็น Virtual Memory อยู่) และอนุญาตให้ Applications อื่น ๆ สามารถมองเห็นเสมือนอยู่ใน address space เดียวกัน ทำให้สามารถ share ข้อมูลที่อยู่ในหน่วยความจำร่วมกันได้ด้วยความเร็วสูง (โดยปกติ Application ที่รันบน Win32 จะแบ่งแยก address space ที่ใช้ออกจากกัน ทำให้ไม่สามารถ share หน่วยความจำกันได้)

หลักการทำงานคงไม่ต้องกล่าวถึงมาก เนื่องจากผมเคยกล่าวถึงบ้างแล้วในบทความชุดก่อนหน้านี้ (แต่ไม่ได้สร้างเป็น Component) สิ่งที่ผมคำนึงถึงในการสร้าง Component ชุดนี้คือพยายามคงความสามารถในการปรับแต่งพารามิเตอร์ต่าง ๆ ให้เปิดกว้างที่สุด เพื่อให้นำไปประยุกต์ใช้งานได้ในหลาย ๆ ด้าน จึงได้สร้างพารามิเตอร์ที่จำเป็นในรูปแบบ Property เพื่อให้สามารถเปลี่ยนแปลงโดยง่าย และใช้งานง่ายที่สุด สำหรับการเข้าถึงข้อมูลผมสร้างกิจกรรมไว้สองกิจกรรมคือ StoreData สำหรับบันทึกข้อมูล และ RetriveData สำหรับอ่านข้อมูล ซึ่งในความเป็นจริงแล้วเราสามารถใช้วิธีการเข้าถึงข้อมูลแบบปกติที่ใช้กันอยู่ในโปรแกรม (ด้วย Pointer) แทนได้เลย เมื่อจบการใช้โปรแกรม ShareMem Component จะทำการยกเลิกหน่วยความจำที่จองไว้ให้เองโดยอัตโนมัติ

สำหรับตัวโปรแกรม ShareMem Component มีส่วนประกอบที่สำคัญใน unit SHAREMEM ดังต่อไปนี้

procedure Alloc(var p : pointer); กิจกรรมสำหรับ allocate หน่วยความจำชนิด Virtual ด้วยขนาดของ dataSize มีขนาดเป็นไบต์ ถ้าสามารถ allocate หน่วยความจำสำเร็จ จะ return ค่า p เป็น pointer ชี้ไปยังหน่วยความจำ โดยจะแจ้งข้อความตอบสนองที่ Notify MessageEvent

procedure MapAlloc(var p : pointer); กิจกรรมสำหรับติดต่อไปยังหน่วยความจำที่ Applications อื่น ๆ สร้างไว้ด้วย aliasName เดียวกัน

function StoreData(p : pointer; size : dword) : boolean; กิจกรรมสำหรับบันทึกข้อมูล p ลง Virtual Memory ที่ allocate ไว้ด้วยขนาด size เป็นไบต์

function RetriveData(var p : pointer; size : dword) : boolean; กิจกรรมสำหรับอ่านข้อมูล ให้กับพอยน์เตอร์ p ด้วยขนาด size FprotectAttr write FprotectAttr;

property dataSize : dword read FmaxSizeLow write FmaxSizeLow; Property สำหรับกำหนดขนาดหน่วยความจำที่ต้องการมีหน่วยเป็นไบต์

property aliasName : shortString read FaliasName write FaliasName; property สำหรับกำหนดชื่อ alias ที่ใช้อ้างถึงหน่วยความจำชุดนี้

property onNotifyMessage : TnotifyMessageEvent read FonNotifyMessage write FonNotifyMessage; เป็น Events ที่ใช้แจ้งลำดับการทำงานใน Component โดยถ้าประมวลผลตามคำสั่งสำเร็จจะให้ค่า MesgCode เป็น 0

โปรแกรมทดสอบ Component

โปรแกรมแสดงดังรูปที่ 9 เป็นโปรแกรมสำหรับทดสอบการใช้งาน ShareMem Component มีส่วนของโปรแกรมที่สำคัญใน unit uTstSM ดังต่อไปนี้

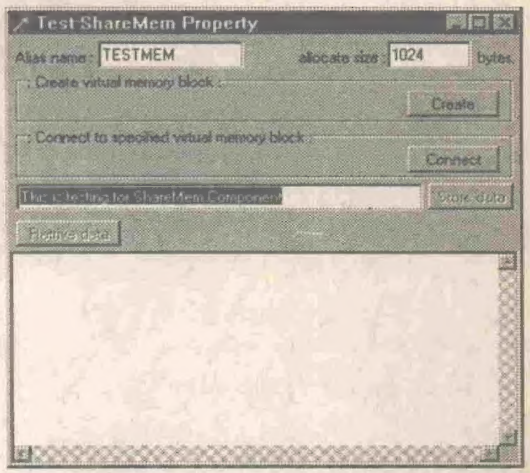
procedure Button1Click(Sender: TObject); กิจกรรมแสดงการ allocate หน่วยความจำ โดยกำหนด aliasName และ dataSize ให้กับ ShareMemC1 Component

procedure Button2Click(Sender: TObject); กิจกรรมแสดงการ Mapping หน่วยความจำที่ address space เดียวกัน โดยการกำหนดชื่อ aliasname เป็นชื่อเดียวกับ application ที่เปิดอยู่ก่อนหน้านี้

procedure Button3Click(Sender: TObject); กิจกรรมสำหรับก๊อบปี้ข้อมูลไปยัง Virtual Memory ที่ allocate ได้

procedure Button4Click(Sender: TObject); กิจกรรมสำหรับก๊อบปี้ข้อมูลจาก Virtual Memory ไปยัง TMemo

procedure ShareMemC1NotifyMessage(sender: TObject; mesgText: ShortString; mesgCode: Integer); กิจกรรมชนิด Events แจ้งข้อความแสดงความสำเร็จในการทำงาน



รูปที่ 9 โปรแกรมทดสอบ ShareMem Component

ส่งท้าย

การทดลองโปรแกรมให้ทำการรันโปรแกรมหลาย ๆ ก๊อบปี้ โดยให้โปรแกรมแรกทำหน้าที่ Create และโปรแกรมที่เหลือทำการ Connect ถ้าการดำเนินการสำเร็จ ไม่ว่าจะเปลี่ยนข้อมูลจากโปรแกรม Application ชุดไหน ข้อมูลของโปรแกรมตัวอื่นก็จะเปลี่ยนตามไปด้วยครับ

จะเห็นว่าการเขียนโปรแกรมชนิด Component ทำได้ไม่ยากเลย แต่ทั้งนี้สิ่งที่ผมกล่าวมาทั้งหมดเป็นเพียงระดับเบื้องต้นเท่านั้น ในการนำไปใช้งานจริงยังจำเป็นต้องศึกษาเพิ่มเติมในส่วน Advance Feature อีกมาก รวมทั้งคงต้องพิจารณาจากชิ้นงานที่ต้องการทำด้วยครับ สวัสดีครับ

***หมายเหตุ** สำหรับซอร์สโค้ดของโปรแกรม ShareMem Component ท่านสามารถใช้บริการซอฟต์เมล์ไมโครฯ ของฉบับนี้*

อิสติงที่ 2 รายการ Components ของโปรแกรมทดสอบ uTSTSM.PAS

```
object Form1: TForm1
  Left = 200
  Top = 108
  BorderStyle = bsSingle
  Caption = 'Test ShareMem Property'
  ClientHeight = 348
  ClientWidth = 370
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 14
  object Label3: TLabel
    Left = 340
    Top = 10
    Width = 28
    Height = 14
    Caption = 'bytes.'
    Transparent = True
  end
  object Label2: TLabel
    Left = 210
    Top = 10
    Width = 64
    Height = 14
    Caption = 'allocate size :'
    Transparent = True
  end
  object Label1: TLabel
    Left = 5
    Top = 10
    Width = 57
    Height = 14
    Caption = 'Alias name :'
    Transparent = True
  end
  object Edit1: TEdit
    Left = 5
    Top = 120
    Width = 296
    Height = 22
    TabOrder = 0
    Text = 'This is testing for ShareMem Component'
  end
  object Button3: TButton
    Left = 305
    Top = 120
    Width = 61
    Height = 21
    Caption = 'Store data'
    Enabled = False
    TabOrder = 1
    OnClick = Button3Click
  end
  object Memo1: TMemo
    Left = 0
    Top = 175
    Width = 370
    Height = 173
    Align = alBottom
    ScrollBars = ssBoth
    TabOrder = 2
  end
  object Button4: TButton
    Left = 5
    Top = 150
```

→

ลิสติงที่ 2 (ต่อ)

```
    Width = 76
    Height = 21
    Caption = 'Retrive data'
    Enabled = False
    TabOrder = 3
    OnClick = Button4Click
  end
  object GroupBox1: TGroupBox
    Left = 5
    Top = 30
    Width = 361
    Height = 41
    Caption = ': Create virtual memory block :'
    TabOrder = 4
    object Button1: TButton
      Left = 285
      Top = 15
      Width = 66
      Height = 21
      Caption = 'Create'
      TabOrder = 0
      OnClick = Button1Click
    end
  end
object Edit2: TEdit
    Left = 65
    Top = 5
    Width = 106
    Height = 22
    TabOrder = 5
    Text = 'TESTMEM'
  end
  object Edit3: TEdit
    Left = 275
    Top = 5
    Width = 61
    Height = 22
    TabOrder = 6
    Text = '1024'
  end
  object GroupBox2: TGroupBox
    Left = 5
    Top = 75
    Width = 361
    Height = 41
    Caption = ': Connect to specified virtual memory block :'
    TabOrder = 7
    object Button2: TButton
      Left = 285
      Top = 15
      Width = 66
      Height = 21
      Caption = 'Connect'
      TabOrder = 0
      OnClick = Button2Click
    end
  end
  object ShareMemC1: TShareMemC
    hFileValue = -1
    protectFlg = SM_PAGE_READWRITE
    protectAttr = SM_SEC_COMMIT
    maxSizeHigh = 0
    maxSizeLow = 0
    dataSize = 0
    mapDesiredAccess = SM_FILE_MAP_ALL_ACCESS
    mapOffsetHigh = 0
    mapOffsetLow = 0
    mapBytesToMap = 0
    onNotifyMessage = ShareMemC1NotifyMessage
    Left = 5
    Top = 180
  end
end
```

เรื่องของ Shift กับ Windows 95 (2)

– เรียก Open With ให้กับไฟล์ที่คุ้นเคย

ถ้าคุณต้องการเปิดไฟล์ที่ถูกกำหนดแล้วว่าต้องเปิดกับโปรแกรมอะไร (associate แล้ว) เช่น เมื่อดับเบิลคลิกไฟล์ Bamboo.bmp เครื่องก็จะเปิดโปรแกรม Paint ให้ทันที แต่ถ้าเราต้องการเปิดไฟล์นี้ด้วยอีกโปรแกรมหนึ่ง ให้กด Shift ค้างไว้ขณะที่คลิกเมาส์ขวา แล้วเมนู Open With จะปรากฏให้คุณได้เลือกใช้ (ปกติจะไม่ปรากฏ) เพื่อเลือกโปรแกรมที่คุณต้องการจะให้เปิดไฟล์นั้น ๆ

– ใช้เมนู SendTo เพื่อความรวดเร็วในการเปิดไฟล์

ถ้าเราเปิด Windows Explorer แล้วต้องการจะแก้ไข Autoexec.bat หรือ config.sys เราจะต้องเปิด Notepad ก่อนแล้วจึงใช้วิธีลากไฟล์จาก Windows Explorer ไปที่ Notepad แล้วจึงจะแก้ไขได้ ดูแล้วเป็นวิธีที่ค่อนข้างจะยุ่งยาก

วิธีทำคือให้ copy ชอร์ตคัตที่ต้องการใช้ เช่น ชอร์ตคัตของ Notepad, Wordpad, Word, Excel, หรือ Printer ก็ได้ครับ ไปที่ \Windows\SendTo เมื่อ copy เสร็จแล้ว เราจะสามารถใช้เมนู SendTo ได้โดยคลิกเมาส์ขวาที่ไฟล์บน Windows Explorer หรือบนเดสก์ท็อปก็ได้แล้วเลือก Send to จะปรากฏโปรแกรมที่เรา copy เข้าไปที่ \Windows\SendTo

เจ้าของเทคนิค
อาวุธ กีรติวงศ์วานิช
599/16 ถ.พรานนก ต.บ้านช่างหล่อ อ.บางกอกน้อย
กรุงเทพฯ 10700
(เจ้าของเทคนิคจะได้รับหนังสือ Delphi จำนวน 1 เล่ม)